Sl/

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/682,140	07/26/2001	Tai-Ying Chiang	SUNP0003USA	4324

27765 7590 07/14/2004

NAIPO (NORTH AMERICA INTERNATIONAL PATENT OFFICE)
P.O. BOX 506
MERRIFIELD, VA 22116

EXAMINER	
MOIDUDDIN, NOREEN	
ART UNIT	**PAPER NUMBER**
2124	5

DATE MAILED: 07/14/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Sc

Office Action Summary	Application No.	Applicant(s)	
	09/682,140	CHIANG ET AL.	
	Examiner	**Art Unit**	
	Noreen Moiduddin	2124	

-- *The MAILING DATE of this communication appears on the cover sheet with the correspondence address* --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *07/26/2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-30* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☐ Claim(s) _____ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *26 July 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED OFFICE ACTION

1. Claims 1-30 were reviewed.

2. Priority date examined: 07/26/2001

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claim 1-4, 11-14, 21-24 rejected under 35 U.S.C. 103(a) as being unpatentable over Clark (U.S. 5,297,150) in view of Shimomura (U.S. 5,854,925).

Claim 1, Clark teaches a method for aiding debugging of program code (testing of programming segments) in a computer system, the method comprising:

(a) **obtaining an error set** of a "bug" or error, **the error set comprising a subset of the program code statements, each program code statement in the error set being relationally connected to the error variable** (Clark, column 4, lines 14-16, column 6, lines 33-35, and lines 36-38). When the application is being tested, the test control program identifies flow paths (program code statements) in an application, which are most likely to contain a "bug" or error (Clark, column 4, lines 14-16). "Once all the searching and ranking (priority values) has occurred, a solution set

(error set) is displayed that indicates the ranked (priority value) flow paths (program code statements) in order (relative ordering) (Clark, column 6, lines 33-35)."

(b) **assigning a priority value to each program code statement in the error set, each priority value indicating a computed probability that the associated program code statement is an error source of the error variable** (Clark, column 3, lines 34 – 40). "Weights (priority values) are assigned to each node (program code statement) based upon the complexity indicator values for conditional and unconditional statements. A determination is then made of a set of flow paths (error set) through the code listing (program code statements), each path ranked in order of assigned weights (computed probability), those flow paths (associated program code statement) with larger assigned weights being assumed as those paths most likely (probable) to fail (error source) (Clark, column 3, lines 34-40)."

(c) **utilizing the output system to present each program code statement in the error set in a manner that indicates the relative ordering of the priority value of the program code statement to the other priority values of the other program code statements in the error set** (Clark, column 6, lines 33-35). "Once all the searching and ranking (priority values) has occurred, a solution set (error set) is displayed (utilizing output system to present) that indicates the ranked (priority value) flow paths (program code statements) in order (relative ordering) (Clark, column 6, lines 33-35).

Clark does not specifically teach **utilizing the input system to indicate an error variable in the program code, the error variable containing an error value that differs from a first desired value,** however he does specify in column 4, lines 14-16, a test control program identifies flow paths (program code statements) in an application

that are most likely to contain a "bug" or error. He also specifies once the most likely paths (instructions) to have malfunctions are displayed, the user proceeds with a node-by-node analysis, and thus, are the paths that are tested first or most thoroughly (column 6, lines 41-45).

Shimomura teaches **utilizing the input system to indicate an error variable in the program code, the error variable containing an error value that differs from a first desired value.** In column 8, lines 5-12, Shimomura teaches a test control part executing an executable program and displaying an input and an output in and from the executable program on a test window. When the programmer selects a wrong part (error part) from the output content with a mouse, the test control part records the output instruction outputting the selected part, the variable name, the value of the variable and the execution point as error information. Such processing is repeated, so that an instruction finally remaining between the normal and abnormal points is determined to be a bug (abstract).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to add the test control part of Shimomura to the test control program of Clark's and use an operator to point out an abnormal point (error variable).

The modification would have been obvious because one of ordinary skill in the art would have been motivated to use the weighted control flow DAG graph created by the test control program of Clark's with the test control part disclosed by Shimomura to find a set of flow paths (code instructions) that are most likely to contain a bug or error (Clark, column 4, lines 14-16). This automates some aspects of software testing, by automatically identifying a set of program paths most susceptible to failure (Clark, column 3, lines 17-20).

As per claim 2, the rejection of claim 1 is incorporated and further, Clark teaches **the method of claim 1 wherein the program code statements in the error set are presented in order from a most: likely error source to a least likely error source** (Clark, column 6, lines 33-35, figure 5, items 74 and 78). "Once all the searching and ranking has occurred, a solution set (the program code statements in the error set) is displayed (presented) that indicates the ranked flow paths (from a most: likely error source to a least likely error source) in order (Clark, column 6, lines 33-35)." In figure 5 of Clark, item 74, paths are ranked in decreasing order of node weights (most likely to least likely).

As per claim 3, the rejection of claim 1 is incorporated and further, Clark teaches **the method of claim 1 wherein each program code statement in the error set is presented with the associated priority value** (Clark, figure 7, column 6, lines 38-40). Figure 7 shows a weighted (priority value) node (nodes represent program code statements) chart representing the information in the knowledge base used internally by a search routine to rank flow paths according to their weights (priority values). In column 6, lines 38-40 Clark specifies that a user may select a particular graphical presentation to display a flow diagram in such a manner that the critical paths are emphasized. "A particular graphical presentation" infers there is more then one graphical presentation available that emphasizes a critical path.

As per claim 4, the rejection of claim 1 is incorporated and further, Clark teaches:

(a) **utilizing the program code statements in** a user specified **set to generate the error set, every program code statement in the error set being in the** specified user set **and also being relationally connected to the error variable**. A user specified set is a search set determined by a predetermined number of flow paths that are found and ranked or a termination condition which is specified by a user (column 6, lines 22-32). A solution set (user specified set) is obtained and critical paths (error set) are highlighted. Critical paths are flow paths and thus specify relational connection to an error variable.

Clark does not teach the use of **a plurality of execution cycles, obtaining an execution set comprising all program code statements that are executed in the error cycle**, and **obtaining an error cycle that is an execution cycle in which the error variable obtains the error value**, although he does specify the use of "prior experience" (column 6, lines 10-12) and the assumption that an application is being tested per each search that identifies the most likely paths containing an error (column 4, lines 13-15). Clark also specifies searches may be terminated when a predetermined number of flow paths have been found and ranked, and to test them before proceeding on to an addition set of flow paths, which specifies the use of more than one execution to obtain an error set (column 6, lines 26-32).

Shimomura teaches the use of **a plurality of execution cycles, obtaining an execution set comprising all program code statements that are executed in the error cycle**, and **obtaining an error cycle that is an execution cycle in which the error variable obtains the error value.** Figure 1, block 11 and column 8, lines 13-20

show an AP rerun part that makes a executable program (plurality of program code statements) rerun up to an abnormal point (point where an error occurred). Figure 2 shows the executable program (plurality of program code statements) is executed two or more times (plurality of execution cycles). The AP rerun part displays a sequence of rerun instructions (execution set) on an execution sequence window (column 8, lines 18-20). Because each rerun sequence runs up to an abnormal point (an abnormal point as specified above in the rejection of claim 1, contains an error variable and error value), any rerun execution sequence (execution set) can be interpreted as an error cycle. During each rerun, a dependence analyzing part (figure 1, block 13) found in an AP rerun part records various types of dependences between the instructions and generates verification-by-division information (column 8, lines 21-28). This information influences which instructions are executed next by AP rerun (column 8, lines 44-47). "Such processing is repeated, so that an instruction finally remaining between the normal and abnormal points is determined to be a bug" (abstract).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to obtain "prior experience" by executing program code a plurality of times to obtain an error cycle, and an execution set as disclosed by Shimomura in Clark's invention to obtain an error set containing the mostly likely cause of an error.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to use "prior experience" as suggested by Clark, to revise nodal weights (priority values). One would also be motivated to use the invention disclosed by Clark to provide a method for program analysis testing that automatically identifies a set of program paths most susceptible to failure (column 3, lines 17-20) in reference to the static slicing referred by Shimomura to map out dependences in the

dependence analyzing part (column 9, lines 45-48, 60-66), that influences instructions executed by the AP rerun part.

As per claim 11, the limitations of claim 11 have already been addressed in connection with claim 1 and 4.

As per claim 12, the rejection of claim 11 is incorporated and further, Clark teaches the method of claim 11 further comprising:

- **assigning a priority value to each statement in the error set, each priority value indicating a computed probability that the associated program code statement in the error set is an error source of the error variable**(Clark, column 3, lines 34 – 40). "Weights (priority values) are assigned to each node (program code statement) based upon the complexity indicator values for conditional and unconditional statements. A determination is then made of a set of flow paths (error set) through the code listing (program code statements), each path ranked in order of assigned weights (computed probability), those flow paths (associated program code statement) with larger assigned weights being assumed as those paths most likely (probable) to fail (error source) (Clark, column 3, lines 34-40)."
- **each program code statement in the error set presented via the output system in a manner that indicates the relative ordering of the priority value of the, program code statement to the other priority values of the other program code statements in the error set.** (Clark, column 6, lines 33-35). "Once all the searching and ranking (priority values) has occurred, a solution set

(error set) is displayed (utilizing output system to present) that indicates the ranked (priority value) flow paths (program code statements) in order (relative ordering) (Clark, column 6, lines 33-35).

As per claim 13, the rejection of claim 12 is incorporated and further, Clark teaches **the method of claim 12 wherein the statements in the error set are presented in order from a most likely error source to a least likely error source** (Clark, column 6, lines 33-35, figure 5, items 74 and 78). "Once all the searching and ranking has occurred, a solution set (the program code statements in the error set) is displayed (presented) that indicates the ranked flow paths (from a most: likely error source to a least likely error source) in order (Clark, column 6, lines 33-35)." In figure 5 of Clark, item 74, paths are ranked in decreasing order of node weights (most likely to least likely).

As per claim 14, the rejection of claim 1 is incorporated and further, Clark teaches **the method of claim 12 wherein each statement in the error set is presented with the associated priority value** (Clark, figure 7, column 6, lines 38-40). Figure 7 shows a weighted (priority value) node (nodes represent program code statements) chart representing the information in the knowledge base used internally by a search routine (the chart is presented internally to the search routine) to rank flow paths according to their weights (priority values). In column 6, lines 38-40 Clark specifies that a user may select a particular graphical presentation to display a flow diagram in such a manner that the critical paths are emphasized. "A particular graphical

presentation" infers there is more then one graphical presentation available that emphasizes a critical path.

As per claim 21, Clark teaches a **computer system comprising:**

- **a processor** (Clark, figure 1, item 30)
- **an output system for presenting information to a user** (Clark, figure 1, item 38)
- **an input system for obtaining data from the user** (Clark, figure 1, item 36)
- **a memory for storing code and data for the processor** (Clark, figure 1, item 34)

the memory comprising: (Clark, figure 1, items 40-47, column 3, lines 68, and column 4, lines 1-20)

- **program code** (application under test) **comprising a plurality of program code statements** (Clark, figure 1, item 44 and 45)
- **debug information** (knowledge base) **about the program code** (Clark, figure 1, item 42, column 4, lines 3)
- **an execution system** (inference engine) **for generating the debug information** (Clark, figure 1, item 43, column 4, lines 3-4)
- **a prioritizing system** (Clark, figure 1, items 40, 45-47, column 4, lines 7-12), **executed by the processor** (Clark, column 3, lines 64-65),
- **that utilizes the debug information to perform the following:**

(a) **obtaining an error set** of a "bug" or error, **the error set comprising a subset of the program code statements, each program code statement in**

the error set being relationally connected to the error variable (Clark, column 4, lines 14-16, column 6, lines 33-35, and lines 36-38). When the application is being tested, the test control program identifies flow paths (program code statements) in an application, which are most likely to contain a "bug" or error (Clark, column 4, lines 14-16). "Once all the searching and ranking (priority values) has occurred, a solution set (error set) is displayed that indicates the ranked (priority value) flow paths (program code statements) in order (relative ordering) (Clark, column 6, lines 33-35)."

(b) **assigning a priority value to each program code statement in the error set, each priority value indicating a computed probability that the associated program code statement is an error source of the error variable** (Clark, column 3, lines 34 – 40). "Weights (priority values) are assigned to each node (program code statement) based upon the complexity indicator values for conditional and unconditional statements. A determination is then made of a set of flow paths (error set) through the code listing (program code statements), each path ranked in order of assigned weights (computed probability), those flow paths (associated program code statement) with larger assigned weights being assumed as those paths most likely (probable) to fail (error source) (Clark, column 3, lines 34-40)."

(c) **utilizing the output system to present each program code statement in the error set in a manner that indicates the relative ordering of the priority value of the program code statement to the other priority values of the other program code statements in the error set** (Clark, column 6, lines 33-35). "Once all the searching and ranking (priority values) has occurred, a

solution set (error set) is displayed (utilizing output system to present) that indicates the ranked (priority value) flow paths (program code statements) in order (relative ordering) (Clark, column 6, lines 33-35).

Clark does not specifically teach **utilizing the input system to indicate an error variable in the program code, the error variable containing an error value that differs from a first desired value,** however he does specify in column 4, lines 14-16, a test control program identifies flow paths (program code statements) in an application that are most likely to contain a "bug" or error. He also specifies once the most likely paths (instructions) to have malfunctions are displayed, the user proceeds with a node-by-node analysis, and thus, are the paths that are tested first or most thoroughly (column 6, lines 41-45).

Shimomura teaches **utilizing the input system to indicate an error variable in the program code, the error variable containing an error value that differs from a first desired value.** In column 8, lines 5-12, Shimomura teaches a test control part executing an executable program and displaying an input and an output in and from the executable program on a test window. When the programmer selects a wrong part (error part) from the output content with a mouse, the test control part records the output instruction outputting the selected part, the variable name, the value of the variable and the execution point as error information. Such processing is repeated, so that an instruction finally remaining between the normal and abnormal points is determined to be a bug (abstract).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to add the test control part of Shimomura to the test

control program of Clark's and use an operator to point out an abnormal point (error variable).

The modification would have been obvious because one of ordinary skill in the art would have been motivated to use the weighted control flow DAG graph created by the test control program of Clark's with the test control part disclosed by Shimomura to find a set of flow paths (code instructions) that are most likely to contain a bug or error (Clark, column 4, lines 14-16). This automates some aspects of software testing, by automatically identifying a set of program paths most susceptible to failure (Clark, column 3, lines 17-20).

As per claim 22, Clark teaches **the computer system of claim 21 wherein the program code statements in the error set are presented in order from a most likely error source to a least likely error source** (Clark, column 6, lines 33-35, figure 5, items 74 and 78). "Once all the searching and ranking has occurred, a solution set (the program code statements in the error set) is displayed (presented) that indicates the ranked flow paths (from a most: likely error source to a least likely error source) in order (Clark, column 6, lines 33-35)." In figure 5 of Clark, item 74, paths are ranked in decreasing order of node weights (most likely to least likely).

As per claim 23, Clark teaches **the computer system of claim 21 wherein each program code statement in the error set is presented with the associated priority value** (Clark, figure 7, column 6, lines 38-40). Figure 7 shows a weighted (priority value) node (nodes represent program code statements) chart representing the information in the knowledge base used internally by a search routine to rank flow paths

according to their weights (priority values). In column 6 of Clark, lines 38-40 Clark specifies that a user may select a particular graphical presentation to display a flow diagram in such a manner that the critical paths are emphasized. "A particular graphical presentation" infers there is more then one graphical presentation available that emphasizes a critical path.

As per claim 24, the rejection of claim 4 is incorporated and further, this claim is rejected for the same reason as noted with the rejection of claim 4.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 5-10, 15-20, and 25-30 rejected under 35 U.S.C. 103(a) as being unpatentable over Clark (U.S. 5,297,150) and Shimomura (U.S. 5,854,925), and further in view of Cuddihy et al (U.S. 5,463,768).

As per claim 5, the rejection of claim 4 is incorporated and further,

As specified in claim 1, Clark and Shimomura teach an error set of nodes (program code statements) is obtained. Clark also specifies in lines 8-12 of column 6,

the use of "prior experience" (prior searching (executing)) in assigning nodal weights (priority values), but Clark does not disclose the method used to assign nodal weights (priority values) based on "prior experience." Neither Clark nor Shimomura teach **obtaining a first sensitized set for the correct variable, the first sensitized set comprising at least a program code statement that is relationally connected to the correct variable, that is executed in the first execution cycle, and that is in the error set.**

Cuddihy et al teaches a method of receiving (**obtaining**) a plurality of error logs (**execution cycles prior to error cycle**) used as a training unit for a new error log (**error cycle**). Referring to table 1 in Cuddihy et al, Cuddihy et al displays eleven historical error logs **executed prior** to a new error log (error cycle). **The first execution cycle** is considered the first executed log listed under column "1". The blocks listed in column "1" are interpreted as the **first sensitized set for the correct variable**. "**The first sensitized set comprising at least a program code statement that is relationally connected to the correct variable, that is executed in the first execution cycle, and that is in the error set"** defines a "metes" condition. **Correct variables** in the context of Cuddihy et al are represented in blocks found in at least two historical error logs. A relationship (**relationally connected**) is established between the blocks (first sensitized set) in the first historical error log (first execution cycle) and the blocks (error set) in a new error log (error cycle). The first historical error log (first execution cycle) in combination with at least one other historical error log (execution cycle) produces a weight value (priority value) used by the new error log (error cycle), including the correct variables. After blocks in each case set (two or more historical error logs) are compared to

identify common blocks (sets). The blocks (elements in a set) are used to characterize fault contribution for a new error log (error cycle) (column 4, lines 54-59).

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to use blocks in a historical error log (first sensitized set of the correct variable) in Clark's and Shimomura's reference to "prior experience".

The modification would have been obvious because one of ordinary skill in the art would have been motivated to use Cuddihy's blocks in a historical error log (first sensitized set) in Clark's and Shimomura's reference to "prior experience", which implies storing prior searches (executions) in memory, to calculate nodal (program code statement) weights (priority values) (Clark column 6, lines 9-11) and to discard weak blocks (program code statements) from being used as diagnostic predictors (Cuddihy column 6, lines 12-14).

Further, Clark and Shimomura do not specifically teach **for each program code statement in the first sensitized set, applying a scaling function to the priority value associated with the identical program code statement in the error set, the scaling function setting a reduced computed probability that the associated program code statement in the program code is the error source of the error variable**. However, Clark and Shimomura do specify "prior experience" can influence nodal weights (program code statements priority values) and that "additional heuristics are contemplated by this invention, but they will vary in accordance with the type and kind of application software being debugged" (Clark, column 5, lines 47-50).

Cuddihy et al teaches a method using historical cases (prior experience) to obtain block (program code statements) weights (priority values) in blocks (program code statements in an error set) in a new error log (error cycle), by blocks (program code statements in an error set) in a new error log (error cycle) with historical cases (prior experience). As specified in part 1 of claim 5's rejection, a historical error log defines an execution cycle and blocks (program code statements) labeled in the log as a sensitized set. A weak block (program code statements) is a block found in all or most of the historical error logs. A weak block represents a least likely (correct variable) block containing a fault (error) in a new error log (error cycle). The weights (**priority value**) of weak blocks (**identical program code statement in the error set**) are assigned a zero value (constant value from a **scaling function**). This reduces the probability (**reduced computed probability**) that it is the main source of the error (column 5, lines 31-34). "Weaker predictors are not allowed to 'out-vote' stronger predictors" (column 5, lines 27-28). Weight values of zero are applied to discard weak blocks (containing correct variables) from being used as a diagnostic predictor for a fault (error) (column 6, lines 14-15).

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to apply a constant (scaling function) to the weight (priority value) associated with the block (identical program code statement) in a new error log (error set logging the results of an error cycle) to the invention disclosed by Clark and Shimomura, where "prior experience" refers to historical error logs and a set of new heuristics refers to the method disclosed by Cuddihy et al.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to discard weak blocks (program code statements) from being used as a

diagnostic predictor (displayed in a critical path-most likely error source) for a fault (error) (column 6, lines 14-15).

As per claim 6, the rejection of claim 5 is incorporated and further, Clark and Shimomura in view of Cuddihy teach:

(a) **the scaling function adds a constant value to the priority value associated with the identical program code statement in the error set** ("The weights (**priority value**) of weak blocks (**identical program code statement in the error set**) are assigned a zero value (constant value from a **scaling function**) (column 5, lines 31-34).")

As per claim 7, the rejection of claim 5 is incorporated and further,

The defined constraints for a second sensitized set do not define a bounding factor. "**The second sensitized set comprising at least a program code statement that is relationally connected to the error variable, that is executed in the first execution cycle, and that is in the error set"** is interpreted as "at least a program code statement" relationally connected to the error variable, "at least a program code statement" executed in the first execution cycle, and "at least a program code statement" in the error set. Because a bounding factor is not provided, a sensitized set can be both a first and a second sensitized set as long as both a correct variable and error variable are relationally connected to at least one program code statement. As specified in the rejection of claim 5, "The first historical error log (first execution cycle) in combination with at least one other historical error log (execution cycle) produces a weight (priority value) used by the new error log (error cycle), including the correct

variables." This statement specifies the blocks contained in the first historical error log (first sensitized set) produce weights used by all the blocks in a new error log (error set), which would include both correct variables and an error variable. Using the weights (priority values) of a historical error log shows a relationship (i.e. relationally connected) between the blocks (sensitized set) of a historical error log and blocks (error set) of a new error log (error cycle containing correct and error variable). Using the same explanation, "**for each program code statement in the second sensitized set, the scaling function is applied to the priority value associated with the identical program code statement in the error set,**" is rejected by the rejection of claim 5. Weak blocks (blocks containing at least one correct variable) are given a weight (priority value) of zero and stronger blocks (blocks containing an error variable) are given a higher weight, as specified in Table 1 of Cuddihy et al.

As per claim 8, the rejection of claim 5 is incorporated and further, Clark teaches **the method of claim 5 wherein a plurality of execution cycles prior to the error cycle are utilized to assign the priority values.**
In lines 8-12 of column 6, Clark specifies nodal (program code statement) weights (priority values) are revised (reassigned), either as a result of prior experience (plurality of execution cycles prior to the error cycle) or as a result of a change in heuristic that indicates a revised value is to be assigned.

As per claim 9, the rejection of claim 5 is incorporated and further, Clark teaches **the method of claim 5 wherein a plurality of correct variables are utilized to assign the priority values.** "A second desired value" is interpreted to mean any value

other than the error value. Given this interpretation, every variable other than an error variable are considered "correct variables". Figure 3, item 52 in Clark says a node value (priority value) is assigned to each basic block (program code statements containing at least one variable). "Each basic blocks" means there is more than one. Figure 6 shows there are more than 3 nodes (program code statements) highlighted in the critical path (most likely error source).

As per claim 10, the rejection of claim 1 is incorporated and further, Clark and Shimomura do not specifically teach **the method of claim 1 wherein the program code is hardware development language (HDL) code**. Clark and Shimomura specifically do not teach adjusting a "weighting factor" of nodes to prevent weights from being greater than a microprocessor's maximum integer, a requirement specified by Cuddihy's et al implementation in a hardware environment, but Clark and Shimomura do teach the use of weights and summed weights to prioritize program code statements.

Cuddihy et al teaches an embodiment of the invention used for hardware implementation (**program code is hardware development language code**) (column 5, lines 41-45). In doing so, Cuddihy specifies the weighting scheme (using weighted values (priority values) in diagnosing a fault) is the integer capability of a typical microprocessor used for hardware implementation of the invention, and therefore the weights should not be greater than the microprocessor's maximum integer. The "weighting factor" (factor used to assign weight values (priority values)) is adjusted, so that it is small enough that the sum of all the weights is within the integer capability of a microprocessor (column 5, lines 45-50).

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to use an adjusted weighting factor described in Cuddihy et al, to implement a hardware environment (program code is hardware development language code) of Clark's and Shimomura's nodal weighting (program code statement priority values) method.

The modification would have been obvious because one of ordinary skill in the art would have been motivated to use an adjusted weighting factor that would prevent the weights from being greater than a microprocessor's maximum integer (column 5, lines 41-50).

As per claims 15-20 and 25-30, these claims are rejected under the same rationale as claims 5-10 respectively.

Conclusion

7. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Noreen Moiduddin whose telephone number is (703) 305-0538. The examiner can normally be reached on Monday, Tuesday, Wednesday, Thursday, and Friday 9:00 AM to 6:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703) 305-9662. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100